Shell Lab

20220124 김문겸

본 LAB에서는 쉘을 시뮬레이션하는 코드를 만드는 것을 목표로 한다. 본 코드에서는 총 7개의 쉘 기능을 구현해야 한다.

1. eval: 입력 받은 command를 parsing하고 interpret하는 쉘의 메인 루틴을 구현해야 한다.
2. builtin_cmd: 입력 받은 command를 interpret하여 대응하는 기능을 실행시켜주는 기능을 구현해야 한다.
3. do_bgfg: bg와 fg 커맨드 기능을 구현해야 한다.
4. waitfg: foreground job이 끝날때까지 shell을 대기시키는 작업을 수행하는 기능을 구현해야 한다.
5. sigchld_handler: SIGCHLD signal을 받았을 때 handling하는 handler의 기능을 구현해야 한다.
6. sigint_handler: SIGINT signal을 받았을 때 handling하는 handler의 기능을 구현해야 한다.
7. sigtstp_handler: SIGTSTP signal을 받았을 때 handling하는 handler의 기능을 구현해야 한다.

eval

먼저 해당 코드를 작성하기 위해, 본 강의의 교안인 CSAPP의 8단원을 참고하였다. 강의 노트 및 교안에 공통적으로 Simple Shell을 구현한 코드가 존재하는데, 해당 코드는 아래와 같다.

```
void eval(char *cmdline)
{

    char *argv[MAXARGS]; /* Argument list execve() */
    char buf[MAXLINE];   /* Holds modified command line */
    int bg;              /* Should the job run in bg or fg? */
    pid_t pid;           /* Process id */
    strcpy(buf, cmdline);
    bg = parseline(buf, argv);
    if (argv[0] == NULL)
        return; /* Ignore empty lines */
    if (!builtin_command(argv))
    {
        if ((pid = Fork()) == 0)
        { /* Child runs user job */
            if (execve(argv[0], argv, environ) < 0)
            {
                printf("%s: Command not found.\n", argv[0]);
                exit(0);
            }
        }
        /* Parent waits for foreground job to terminate */
        if (!bg)
        {
            int status;
            if (waitpid(pid, &status, 0) < 0)
                unix_error("waitfg: waitpid error");
        }
        else
            printf("%d %s", pid, cmdline);
    }
    return;
}
```

위 코드는 shell의 구조만을 표현하기 위한 단순한 코드이다. 따라서 이번 LAB에서 요구하는 기본적인 기능을 탑재하기 위해 코드를 추가 및 수정해야 한다.

가장 먼저 writeup 파일의 HINT에 제공된 요구사항들을 구현하였다. shell이 자식 프로세스를 만드는 과정에서, addjob 함수 호출 전 자식 프로세스가 SIGCHLD에 의해 reaping되는 것을 방지하기 위해, SIGCHLD 시그널을 Block해야만 한

다. 하지만 Block만 해놓고 그대로 둔다면 자식 프로세스가 부모의 Block vector를 상속받기 때문에, 자식 프로세스에서 프로그램을 실행하기 전(execve) block해 둔 SIGCHLD 시그널을 unblock했다.

또한 HINT에서 요구한 setpgid(0, 0)을 fork 후 실행 전에 기입해주었다.

```
if(!builtin_cmd(argv)){

    /*According to Hint*/
    sigemptyset(&block_vector);
    sigaddset(&block_vector, SIGCHLD);
    sigprocmask(SIG_BLOCK, &block_vector, NULL); // SIGCHLD signal blocking

    if((pid = fork()) == 0){

        setpgid(0, 0);
        sigprocmask(SIG_UNBLOCK, &block_vector, NULL);

        if(execve(argv[0], argv, environ) < 0){
```

위 코드에서는 builtin_cmd의 결과가 0이라면 위의 코드를 실행하도록 설계되었다.builtin_cmd가 1이라면 빌트인 명령어임을, 0이면 아닌 것을 의미한다. 이후에 설명할 builtin_cmd에서 builtin 커맨드인 quit, bg, fg, jobs의 경우 1을 리턴하고, 아닌 경우 0을 리턴하도록 설계했다.
자식 프로세스의 block vector을 execve 실행 전 unblock하여 사고를 방지한다. 부모 프로세스의 경우에도 마찬가지로 이전에 block해 둔 SIGCHLD를 Unblock해주어야 하므로 아래와 같이 코드를 작성해준다.

```
if(!bg){
    addjob(jobs, pid, FG, cmdline);
    sigprocmask(SIG_UNBLOCK, &block_vector, NULL); // unblocking
    waitfg(pid); //reaping process by using 'waitfg'
}
else{
    addjob(jobs, pid, BG, cmdline); // calling addjob
    sigprocmask(SIG_UNBLOCK, &block_vector, NULL); // unblocking
    printf("[%d] (%d) %s",pid2jid(pid), (int)pid, cmdline);
```

builtin_cmd

builtin_cmd에서는 command의 첫 번째 문자열에 들어오는 명령어를 인식하여 각각의 기능을 수행하도록 분류해준다.

```
if(!strcmp(argv[0], "quit")){
    exit(0);
}

if(!strcmp(argv[0], "fg")){
    do_bgfg(argv);
    return 1;
}

if(!strcmp(argv[0], "bg")){
    do_bgfg(argv);
    return 1;
}

if(!strcmp(argv[0], "jobs")){
    listjobs(jobs);
    return 1;
}
```

quit일 경우 exit하여 종료하고, fg일 경우 해당 기능을 처리하는 do_bgfg 함수를 불러오고, 1을 반환하며 종료한다. bg의 경우도 마찬가지이다. jobs의 경우 현재 존재하는 job들을 모두 print한다.

```
    return 0;     /* not a b
}
```

만약 builtin command(quit, fg, bg, jobs)가 아니라면, 0을 반환한다. 이는 위의 eval 함수 구현에서 간접적으로 언급된 바 있다.

do_bgfg

do_bgfg 함수를 구현하기 전에, bg와 fg의 대한 기능을 간략하게 알아보자면,
bg - 중지된 background job을 실행 중인 background job으로 바꾼다.
fg - 중지되었거나 실행중인 background job을 실행중인 foreground job으로 바꾼다.

본 함수 구현에서는 error handling이 요구되었다. 주어진 tshref.out 파일의 test14 result를 참고하여, 에러를 처리해주었다.

가장 먼저 "tsh> fg"와 같이 bg/fg 명령어 입력 후 뒤의 job의 pid/jid를 기입하지 않은 경우의 에러를 핸들링해주었다.

```
if(argv[1] == NULL){ //error handling: ex) fg
    if(!strcmp(argv[0], "fg")){
        printf("fg command requires PID or %%jobid argument\n");
    }
    else if(!strcmp(argv[0], "bg")){
        printf("bg command requires PID or %%jobid argument\n");
    }
    return;
}
```

fg/bg 명령어와 함께 기입되는 'job'에는 해당 job의 pid 혹은 jid(job id)가 들어갈 수 있다. 이때 두 종류를 구분하기 위해 pid는 단순 숫자를, jid는 앞에 '%' 기호를 붙인다. 앞서 job_num이라는 int형 변수를 선언하여, pid, jid의 숫자를 저장해

주는 작업을 진행하였다. 이때 jid의 경우 앞의 '%' 기호를 없애야 하므로 조건문을 통해 추가적인 작업을 수행해주었다.

```
if(argv[1][0] == '%'){ // JID

int i;
for(i=1;argv[1][i] != '\0';i++){
    temp[i-1] = argv[1][i];
    if(!isdigit(argv[1][i])){
        nodgt_check = 1;
    }
}
job_num = atoi(temp);
jid_check = 1;
```

여기서 추가적으로 error handling을 하기 위한 코드를 추가했다. 위 코드를 보면 'ctype.h' 헤더파일에 있는 isdigit 함수를 이용해서, job으로 입력받은 문자열에 integer가 아닌 char이 있는지 확인하여 있다면 nodgt_check를 1로 체크하는 기능이 구현되어 있다. 즉, 정상적인 pid나 jid의 경우 숫자값을 가지는데, 'fg %a'와 같이 문자 pid/jid를 입력할 경우 정상적이지 않은 과정이므로 에러 처리를 해주어야 한다. 따라서 아래 코드에서는 그러한 경우에(nodgt_check == 1) 에러 메세지를 출력하도록 구현한 것을 확인할 수 있다.

```
    if(nodgt_check == 1){ //error handling: ex) fg %a
    if(!strcmp(argv[0], "fg")){
        printf("fg: argument must be a PID or %%jobid\n");
    }
    else if(!strcmp(argv[0], "bg")){
        printf("bg: argument must be a PID or %%jobid\n");
    }
    return;
}
```

숫자 값의 pid/jid를 입력 받았다고 해도, 해당 값을 pid/jid로 가지는 job이 존재하지 않다면 그것 또한 오류를 범한다. 따라서 그러한 경우에 job이 없다는 에러 메세지를 출력하도록 한다.

```
if(getjobjid(jobs,job_num) == NULL){ //error handling: ex) fg %9999
    if(!strcmp(argv[0], "fg")){
        printf("fg: No such job\n");
    }
    else if(!strcmp(argv[0], "bg")){
        printf("bg: No such job\n");
    }
    return;
}
```

PID 입력의 경우에도 똑같이 에러를 처리해준다.

```
else{ // PID

    int i;
    for(i=0;argv[1][i] != '\0';i++){
    temp[i] = argv[1][i];
    if(!isdigit(argv[1][i])){
        nodgt_check = 1;
        }
    }
    job_num = atoi(argv[1]);
    jid_check = 0;

    if(nodgt_check == 1){//error handling: ex) fg a
        if(!strcmp(argv[0], "fg")){
            printf("fg: argument must be a PID or %%jobid\n");
        }
        else if(!strcmp(argv[0], "bg")){
            printf("bg: argument must be a PID or %%jobid\n");
        }
        return;
    }


    if(getjobpid(jobs,job_num) == NULL){ //error handling: ex) fg 9999
        printf("(%d): No such process\n", job_num);
        return;
    }
}
```

앞선 과정을 통해 에러 처리 및 fg/bg와 함께 입력된 pid/jid를 job_num 변수에 저장하였다. 이제 job_num 변수에 저장된 pid/jid 값을 가지고 해당되는 job을 찾아야 한다. 앞에서 argv[1][0]이 %라면, 즉 jid라면 jid_check를 1로, 아니라면, 즉 pid라면 pid_check = 0으로 세팅해주었다. 이에 따라서 각각 getjobjid, getjobpid를 실행하여 해당 pid/jid를 가진 job을 불러온다.

이제 bg/fg에 대한 본격적인 기능을 수행하는 코드를 작성한다. 우선 bg/fg의 기능을 수행하기 위해서 각 job에 SIGCONT 시그널을 보내어 job을 다시 시작하게 해야 한다. 따라서 kill 함수를 통해 SIGCONT 시그널을 각 프로세스에 보낸다.

```
if(jid_check == 1){ // JID
    newjob = getjobjid(jobs, job_num);
}
else{
    newjob = getjobpid(jobs, job_num);
}

kill(-newjob->pid, SIGCONT);
```

이제 fg/bg에 따른 각각의 기능들을 수행한다. fg의 경우 job의 상태를 ForeGround로 변경하고, 해당 작업이 reaping될 때까지 waitfg를 통해 wait한다. 이는 foreground의 특성에 따른 구현이다. bg일 경우, job의 상태를 BackGround로 설정하고, tshref.out 파일에 따라 bg 명령어 실행 후 해당 프로세스의 jid, pid 및 command line을 출력한다.

```
if(!strcmp(argv[0], "fg")){
    newjob->state = FG;
    waitfg(newjob->pid);
}

if(!strcmp(argv[0], "bg")){
    newjob->state = BG;
    printf("[%d] (%d) %s", newjob->jid, newjob->pid, newjob->cmdline);

}
```

waitfg

waitfg는 프로세스가 종료될 때까지 대기하는 기능을 수행하는 함수이다. 이를 위해, writeup 파일에서는 waitpid 함수 대신 sleep으로 구현하는 것을 권장하고 있기 때문에 sleep으로 구현하였다.

```
void sigint_handler(int sig)
{
    pid_t pid_num = fgpid(jobs);
    if(pid_num != 0){
    kill(-pid_num, sig);
    }
    return;
}
```

현재 foreground에서 실행 중인 job의 pid가 argument로 받은 pid와 같을 때까지 sleep(1)을 실행한다. sleep(n) 함수는 n초 동안 대기하는 함수이다. 만약 foreground에서 실행되던 프로세스가 종료된다면, fgpid(jobs)는 0을 반환하므로 while loop에서 빠져나오고 함수가 종료된다. 즉, 이 함수를 통해 foreground의 작업이 모두 완료될 때까지 다른 작업의 수행을 막고 대기하는 기능을 수행한다.

sigchld_handler

sigchld_handler는 SIGCHLD 시그널이 발생할 때 해당 시그널을 처리하는 함수이다. SIGCHLD는 자식 프로세스가 종료 및 중지 될때마다 셸에 전송되는 신호이다. 주어진 주석에 따라, 좀비 프로세스들을 모두 reaping한다. 이때 현재 실행 중인 다른 모든 종료될 때까지 기다리지 않는다.

반복문이 반복될 때마다 waitpid를 진행시켜 좀비 프로세스를 reaping시킨다. 이때 waitpid의 option을 WNOHANG|WUNTRACED로 사용한다. 이는 프로세스가 모두 종료되었다면 0을 즉시 리턴하는 WNOHANG과, 정지 및 종료 상태의 PID를 리턴하는 WUNTRACED를 모두 이용함을 의미한다. 따라서 이 덕분에 반복문마다 정지 및 종료 상태의 프로세스를 reaping시키고 해당 프로세스의 pid를 얻고, 만약 정지 및 종료 된 자식들이 모두 처리 된 후 더 이상 남아있지 않다면 0을 반환하고, 호출한 프로세스에 자식이 없다면 -1을 리턴한다. 이러한 0과 -1의 리턴을 받는 경우, 더 이상 handling을 하지 않아도 되므로 while loop를 벗어나 handler을 종료한다.

SIGCHLD의 경우 stop과 terminate가 발생하면 호출되는 시그널이므로, stop의 경우 해당 시그널을 호출한 job의 state를 stop을 의미하는 ST로 변경하고, terminate의 경우 해당 job을 삭제한다. 이때 시그널을 전송받아 종료될 경우 tshref.out 파일에서 메세지 출력을 동반하므로, 시그널에 의한 프로세스의 종료를 인식하는 WIFSIGNALED가 true라면 메세지 + deletejob을 수행하고, quit과 같이 시그널이 아닌 단순한 종료의 경우 메세지 없이 deletejob을 수행한다.

```
while(1){
    pid_num = waitpid(-1, &statusp, WNOHANG | WUNTRACED);
    if(pid_num == -1 || pid_num == 0){
        break;
    }
    else{
        newjob = getjobpid(jobs, pid_num);
        if(WIFSIGNALED(statusp)){
            printf("Job [%d] (%d) terminated by signal 2\n", newjob->jid, newjob->pid);
            deletejob(jobs, pid_num);
        }
        else if(WIFEXITED(statusp)){ // for quit: there's no printed information
            deletejob(jobs, pid_num);
        }
        else if(WIFSTOPPED(statusp)){
            printf("Job [%d] (%d) stopped by signal 20\n", newjob->jid, newjob->pid);
            newjob->state = ST;
        }
    }
}
```

sigint_handler

```
void sigint_handler(int sig)
{
    pid_t pid_num = fgpid(jobs);
    if(pid_num > 0){
    kill(-pid_num, sig);
    }
    return;
}
```

sigint_handler는 SIGINT 시그널을 받았을 때 처리하는 handler이다. SIGINT가 발생하면, 현재 foreground에서 실행중인 job의 pid를 얻은 후 kill 함수를 통해 해당 pid를 갖는 job에 시그널을 보낸다. pid_num의 경우 항상 0보다 크므로 조건문을 통해 처리해준다. 해당 조건문이 없을 경우 job이 없는 초기 상태에서 Ctrl+C를 누를 경우 에러가 발생한다.

sigtstp_handler

```
void sigtstp_handler(int sig)
{
    pid_t pid_num = fgpid(jobs);

    if(pid_num > 0){
        kill(-pid_num, sig);
    }

    return;
}
```

sigint_handler와 마찬가지로 SIGTSTP 시그널이 발생하면 foreground에서 실행중인 job의 id를 얻은 후 kill 함수를 통해 시그널 처리를 해주었다.

Error handling

코드 작성 후 보고서를 작성하던 와중 fork, kill 등의 함수에 대한 시스템 에러를 처리해야 한다는 것을 알게 되어, 불가피하게 별도로 에러 처리를 위해 추가한 코드에 대한 설명을 작성했다.

```
if (sigemptyset(&block_vector) != 0)
{
    unix_error("sigemptyset error");
}
if (sigaddset(&block_vector, SIGCHLD) != 0)
{
    unix_error("sigaddset error");
}
if (sigprocmask(SIG_BLOCK, &block_vector, NULL) != 0)
{
    unix_error("sigprocmask error");
} // SIGCHLD signal blocking
```

먼저 sigemptyset, sigaddset, sigprocmask에 대해 에러 처리해주었다.

```
pid = fork();
if(pid == -1){
    unix_error("fork error");
}

if(pid == 0){
```

fork()의 경우 에러 발생시 -1을 반환한다. 이때 pid = fork()를 먼저 실행한 후, pid값에 따라 조건문을 분할하였다. 만약 조건문 안에 "pid=fork()"라는 구문이 모두 들어가게 되면, 자식 프로세스가 두번 생기기 때문이다.

```
setpgid(0, 0);
if (sigprocmask(SIG_UNBLOCK, &block_vector, NULL) != 0)
{
    unix_error("sigprocmask error");
}
```

이 밖에도 eval 함수 내부의 sigprocmask에 대해서 모두 에러 처리를 해주었다.

```
if(kill(-newjob->pid, SIGCONT) == -1){
    unix_error("to_bgfg error");
}
```

do_bgfg 함수 내부에서 실행되는 kill 함수도 에러가 발생할 경우 -1을 반환하므로, 에러처리해준다.

```
void sigint_handler(int sig)
{
    pid_t pid_num = fgpid(jobs);
    if(pid_num > 0){
        if(kill(-pid_num, sig) == -1){
            unix_error("sigint error");
        }
    }
    return;
}

/*
 * sigtstp_handler - The kernel sends a SIGTSTP t
 *     the user types ctrl-z at the keyboard. Cat
 *     foreground job by sending it a SIGTSTP.
 */
void sigtstp_handler(int sig)
{
    pid_t pid_num = fgpid(jobs);

    if(pid_num > 0){
        if(kill(-pid_num, sig) == -1){
            unix_error("sigtstp error");
        }
    }
    return;
}
```

sigint_handler와 sigtstp_handler 함수에도 kill이 있으므로, 에러처리해준다.

test

test01

```
bash-4.2$ make test01
./sdriver.pl -t trace01.txt -s ./tsh -a "-p"
#
# trace01.txt - Properly terminate on EOF.
#
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace01.txt -s ./tsh -a "-p"
#
# trace01.txt - Properly terminate on EOF.
#
```

test02

```
bash-4.2$ make test02
./sdriver.pl -t trace02.txt -s ./tsh -a "-p"
#
# trace02.txt - Process builtin quit command.
#
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace02.txt -s ./tsh -a "-p"
#
# trace02.txt - Process builtin quit command.
#
```

test03

```
bash-4.2$ make test03
./sdriver.pl -t trace03.txt -s ./tsh -a "-p"
#
# trace03.txt - Run a foreground job.
#
tsh> quit
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace03.txt -s ./tsh -a "-p"
#
# trace03.txt - Run a foreground job.
#
tsh> quit
```

test04

```
bash-4.2$ make test04
./sdriver.pl -t trace04.txt -s ./tsh -a "-p"
#
# trace04.txt - Run a background job.
#
tsh> ./myspin 1 &
[1] (14957) ./myspin 1 &
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace04.txt -s ./tsh -a "-p"
#
# trace04.txt - Run a background job.
#
tsh> ./myspin 1 &
[1] (26252) ./myspin 1 &
```

test05

```
bash-4.2$ make test05
./sdriver.pl -t trace05.txt -s ./tsh -a "-p"
#
# trace05.txt - Process jobs builtin command.
#
tsh> ./myspin 2 &
[1] (15019) ./myspin 2 &
tsh> ./myspin 3 &
[2] (15022) ./myspin 3 &
tsh> jobs
[1] (15019) Running ./myspin 2 &
[2] (15022) Running ./myspin 3 &
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace05.txt -s ./tsh -a "-p"
#
# trace05.txt - Process jobs builtin command.
#
tsh> ./myspin 2 &
[1] (26256) ./myspin 2 &
tsh> ./myspin 3 &
[2] (26258) ./myspin 3 &
tsh> jobs
[1] (26256) Running ./myspin 2 &
[2] (26258) Running ./myspin 3 &
```

test06

```
bash-4.2$ make test06
./sdriver.pl -t trace06.txt -s ./tsh -a "-p"
#
# trace06.txt - Forward SIGINT to foreground job.
#
tsh> ./myspin 4
Job [1] (15253) terminated by signal 2
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace06.txt -s ./tsh -a "-p"
#
# trace06.txt - Forward SIGINT to foreground job.
#
tsh> ./myspin 4
Job [1] (26263) terminated by signal 2
```

test07

```
^[bash-4.2$ make test07
./sdriver.pl -t trace07.txt -s ./tsh -a "-p"
#
# trace07.txt - Forward SIGINT only to foreground job.
#
tsh> ./myspin 4 &
[1] (15671) ./myspin 4 &
tsh> ./myspin 5
Job [2] (15674) terminated by signal 2
tsh> jobs
[1] (15671) Running ./myspin 4 &
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace07.txt -s ./tsh -a "-p"
#
# trace07.txt - Forward SIGINT only to foreground job.
#
tsh> ./myspin 4 &
[1] (26267) ./myspin 4 &
tsh> ./myspin 5
Job [2] (26269) terminated by signal 2
tsh> jobs
[1] (26267) Running ./myspin 4 &
```

test08

```
bash-4.2$ make test08
./sdriver.pl -t trace08.txt -s ./tsh -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (15798) ./myspin 4 &
tsh> ./myspin 5
Job [2] (15801) stopped by signal 20
tsh> jobs
[1] (15798) Running ./myspin 4 &
[2] (15801) Stopped ./myspin 5
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace08.txt -s ./tsh -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (26274) ./myspin 4 &
tsh> ./myspin 5
Job [2] (26276) stopped by signal 20
tsh> jobs
[1] (26274) Running ./myspin 4 &
[2] (26276) Stopped ./myspin 5
```

test09

```
bash-4.2$ make test09
./sdriver.pl -t trace09.txt -s ./tsh -a "-p"
#
# trace09.txt - Process bg builtin command
#
tsh> ./myspin 4 &
[1] (15906) ./myspin 4 &
tsh> ./myspin 5
Job [2] (15910) stopped by signal 20
tsh> jobs
[1] (15906) Running ./myspin 4 &
[2] (15910) Stopped ./myspin 5
tsh> bg %2
[2] (15910) ./myspin 5
tsh> jobs
[1] (15906) Running ./myspin 4 &
[2] (15910) Running ./myspin 5
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace08.txt -s ./tsh -a "-p"
#
# trace08.txt - Forward SIGTSTP only to foreground job.
#
tsh> ./myspin 4 &
[1] (26274) ./myspin 4 &
tsh> ./myspin 5
Job [2] (26276) stopped by signal 20
tsh> jobs
[1] (26274) Running ./myspin 4 &
[2] (26276) Stopped ./myspin 5
```

test10

```
bash-4.2$ make test10
./sdriver.pl -t trace10.txt -s ./tsh -a "-p"
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
[1] (16026) ./myspin 4 &
tsh> fg %1
Job [1] (16026) stopped by signal 20
tsh> jobs
[1] (16026) Stopped ./myspin 4 &
tsh> fg %1
tsh> jobs
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace10.txt -s ./tsh -a "-p"
#
# trace10.txt - Process fg builtin command.
#
tsh> ./myspin 4 &
[1] (26290) ./myspin 4 &
tsh> fg %1
Job [1] (26290) stopped by signal 20
tsh> jobs
[1] (26290) Stopped ./myspin 4 &
tsh> fg %1
tsh> jobs
```

test11

```
bash-4.2$ make test11
./sdriver.pl -t trace11.txt -s ./tsh -a "-p"
#
# trace11.txt - Forward SIGINT to every process in foreground process group
#
tsh> ./mysplit 4
Job [1] (20713) terminated by signal 2
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
 1048 pts/167  Ss+    0:00 -bash
 1277 pts/27   Ss+    0:00 /bin/bash --init-file /home/std/leejiwon1125/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad
d9fd269ab1884/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 1567 tty1     Ss+    0:00 -bash
 1926 pts/142  Ss+    0:00 /bin/bash --init-file /home/std/kihyun/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb66
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 2184 pts/149  Ss+    0:00 -bash
 2291 pts/149  R      8:00 ./tsh -p
 3652 pts/66   Ss     0:00 /bin/bash --init-file /home/std/kkomy/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666
debe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 3926 pts/116  S+     0:00 /tmp/.mount_nvim.atFiUct/usr/bin/nvim tsh.c
 4387 pts/150  S+     0:00 vim tsh.c
 4447 pts/135  T      0:00 make test10
 4451 pts/135  T      0:00 /usr/bin/perl ./sdriver.pl -t trace10.txt -s ./tsh -a -p
 4454 pts/135  S      0:00 ./tsh -p
 5205 pts/96   Ss+    0:00 /bin/bash --init-file /home/std/imhakyeum/.vscode-server/bin/1a5daa3a0231a0fbba4f14db7ec46
cf99d7768e/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 5982 pts/125  Ss+    0:00 -bash
 6144 pts/104  Ss+    0:00 /bin/bash --init-file /home/std/kimyeonjin/.vscode-server/bin/af28b32d7e553898b2a91af498b1
b666fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 6707 pts/106  Ss+    0:00 -bash
 6715 pts/109  Ss+    0:00 -bash
 7525 pts/60   Ss+    0:00 /bin/bash --init-file /home/std/yen306/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb66
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
```

rtest11의 결과와 동일한 결과를 출력했다.

```
bash-4.2$ make rtest11
./sdriver.pl -t trace11.txt -s ./tshref -a "-p"
#
# trace11.txt - Forward SIGINT to every process in foreground process group
#
tsh> ./mysplit 4
Job [1] (20325) terminated by signal 2
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
 1048 pts/167  Ss+    0:00 -bash
 1277 pts/27   Ss+    0:00 /bin/bash --init-file /home/std/leejiwon1125/.vscode-server/bin/ee2b180d582a7f601fa6ecfd
d9fd269ab1884/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 1567 tty1     Ss+    0:00 -bash
 1926 pts/142  Ss+    0:00 /bin/bash --init-file /home/std/kihyun/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 2184 pts/149  Ss+    0:00 -bash
 2291 pts/149  R      7:57 ./tsh -p
 3652 pts/66   Ss     0:00 /bin/bash --init-file /home/std/kkomy/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb6
debe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 3926 pts/116  S+     0:00 /tmp/.mount_nvim.atFiUct/usr/bin/nvim tsh.c
 4387 pts/150  S+     0:00 vim tsh.c
 4447 pts/135  T      0:00 make test10
 4451 pts/135  T      0:00 /usr/bin/perl ./sdriver.pl -t trace10.txt -s ./tsh -a -p
 4454 pts/135  S      0:00 ./tsh -p
 5205 pts/96   Ss+    0:00 /bin/bash --init-file /home/std/imhakyeum/.vscode-server/bin/1a5daa3a0231a0fbba4f14db7ec
cf99d7768e/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 5982 pts/125  Ss+    0:00 -bash
 6144 pts/104  Ss+    0:00 /bin/bash --init-file /home/std/kimyeonjin/.vscode-server/bin/af28b32d7e553898b2a91af498
b666fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 6707 pts/106  Ss+    0:00 -bash
 6715 pts/109  Ss+    0:00 /bin/bash --init-file /home/std/yen306/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
```

test12

```
bash-4.2$ make test12
./sdriver.pl -t trace12.txt -s ./tsh -a "-p"
#
# trace12.txt - Forward SIGTSTP to every process in foreground process group
#
tsh> ./mysplit 4
Job [1] (21206) stopped by signal 20
tsh> jobs
[1] (21206) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
 1048 pts/167  Ss+    0:00 -bash
 1277 pts/27   Ss+    0:00 /bin/bash --init-file /home/std/leejiwon1125/.vscode-server/bin/ee2b180d582a7f601fa6e
d9fd269ab1884/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 1567 tty1     Ss+    0:00 -bash
 1926 pts/142  Ss+    0:00 /bin/bash --init-file /home/std/kihyun/.vscode-server/bin/af28b32d7e553898b2a91af498b
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 2184 pts/149  Ss+    0:00 -bash
 2291 pts/149  R      8:03 ./tsh -p
 3652 pts/66   Ss     0:00 /bin/bash --init-file /home/std/kkomy/.vscode-server/bin/af28b32d7e553898b2a91af498b1
debe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 3926 pts/116  S+     0:00 /tmp/.mount_nvim.atFiUct/usr/bin/nvim tsh.c
 4387 pts/150  S+     0:00 vim tsh.c
 4447 pts/135  T      0:00 make test10
 4451 pts/135  T      0:00 /usr/bin/perl ./sdriver.pl -t trace10.txt -s ./tsh -a -p
 4454 pts/135  S      0:00 ./tsh -p
 5205 pts/96   Ss+    0:00 /bin/bash --init-file /home/std/imhakyeum/.vscode-server/bin/1a5daa3a0231a0fbba4f14db
cf99d7768e/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 5982 pts/125  Ss+    0:00 -bash
 6144 pts/104  Ss+    0:00 /bin/bash --init-file /home/std/kimyeonjin/.vscode-server/bin/af28b32d7e553898b2a91af
b666fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 6707 pts/106  Ss+    0:00 -bash
 6715 pts/109  Ss+    0:00 -bash
 7525 pts/60   Ss+    0:00 /bin/bash --init-file /home/std/yen306/.vscode-server/bin/af28b32d7e553898b2a91af498b
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
```

rtest12의 결과와 동일한 결과를 출력했다.

```
bash-4.2$ make rtest12
./sdriver.pl -t trace12.txt -s ./tshref -a "-p"
#
# trace12.txt - Forward SIGTSTP to every process in foreground process group
#
tsh> ./mysplit 4
Job [1] (21382) stopped by signal 20
tsh> jobs
[1] (21382) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
 1048 pts/167  Ss+    0:00 -bash
 1277 pts/27   Ss+    0:00 /bin/bash --init-file /home/std/leejiwon1125/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8
d9fd269ab1884/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 1567 tty1     Ss+    0:00 -bash
 1926 pts/142  Ss+    0:00 /bin/bash --init-file /home/std/kihyun/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 2184 pts/149  Ss+    0:00 -bash
 2291 pts/149  R      8:04 ./tsh -p
 3652 pts/66   Ss     0:00 /bin/bash --init-file /home/std/kkomy/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666f
debe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 3926 pts/116  S+     0:00 /tmp/.mount_nvim.atFiUct/usr/bin/nvim tsh.c
 4387 pts/150  S+     0:00 vim tsh.c
 4447 pts/135  T      0:00 make test10
 4451 pts/135  T      0:00 /usr/bin/perl ./sdriver.pl -t trace10.txt -s ./tsh -a -p
 4454 pts/135  S      0:00 ./tsh -p
 5205 pts/96   Ss+    0:00 /bin/bash --init-file /home/std/imhakyeum/.vscode-server/bin/1a5daa3a0231a0fbba4f14db7ec463
cf99d7768e/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 5982 pts/125  Ss+    0:00 -bash
 6144 pts/104  Ss+    0:00 /bin/bash --init-file /home/std/kimyeonjin/.vscode-server/bin/af28b32d7e553898b2a91af498b1f
b666fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 6707 pts/106  Ss+    0:00 -bash
 6715 pts/109  Ss     0:00 -bash
 7525 pts/60   Ss+    0:00 /bin/bash --init-file /home/std/yen306/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 8590 pts/46   Ss+    0:00 -bash
```

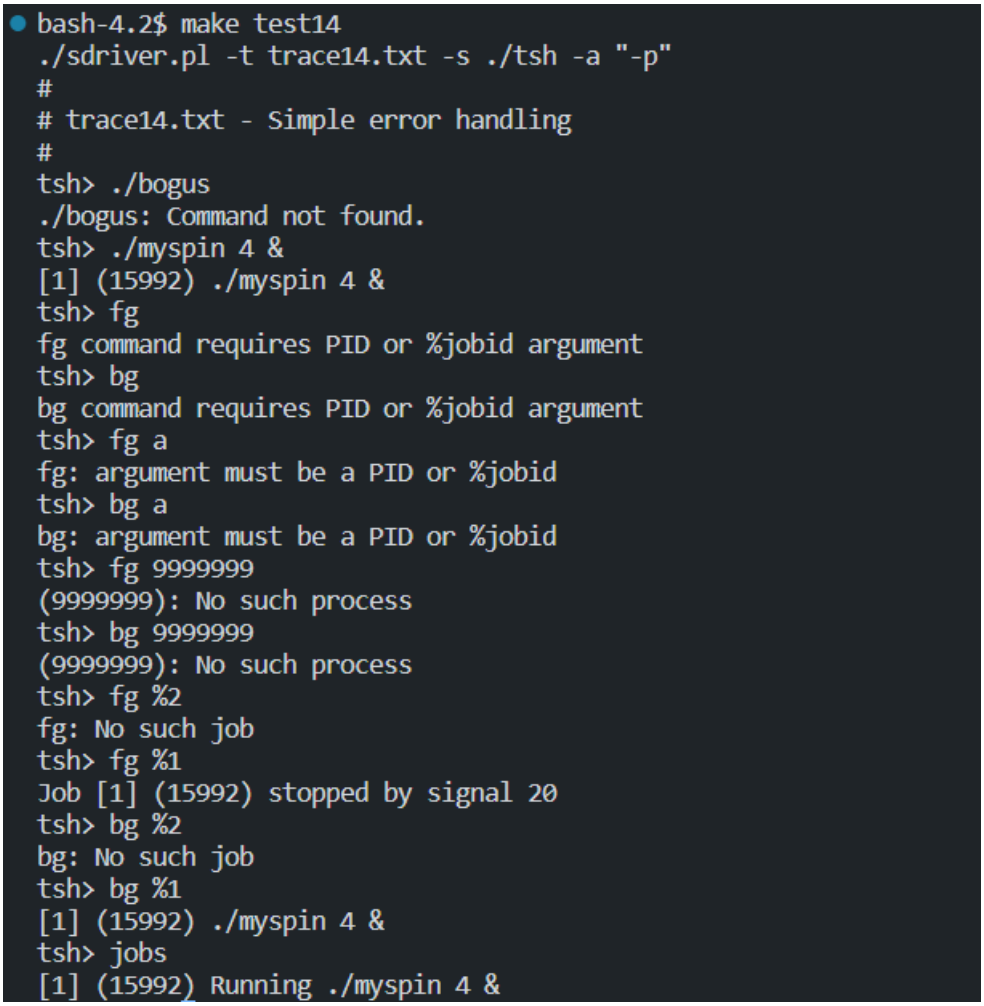
test13

```
bash-4.2$ make test13
./sdriver.pl -t trace13.txt -s ./tsh -a "-p"
#
# trace13.txt - Restart every stopped process in process group
#
tsh> ./mysplit 4
Job [1] (21677) stopped by signal 20
tsh> jobs
[1] (21677) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
 1048 pts/167  Ss+    0:00 -bash
 1277 pts/27   Ss+    0:00 /bin/bash --init-file /home/std/leejiwon1125/.vscode-server/bin/ee2b180d582a7f601fa6ecfd
d9fd269ab1884/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 1567 tty1     Ss+    0:00 -bash
 1926 pts/142  Ss+    0:00 /bin/bash --init-file /home/std/kihyun/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 2184 pts/149  Ss+    0:00 -bash
 2291 pts/149  R      8:07 ./tsh -p
 3652 pts/66   Ss     0:00 /bin/bash --init-file /home/std/kkomy/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb6
debe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 3926 pts/116  S+     0:00 /tmp/.mount_nvim.atFiUct/usr/bin/nvim tsh.c
 4387 pts/150  S+     0:00 vim tsh.c
 4447 pts/135  T      0:00 make test10
 4451 pts/135  T      0:00 /usr/bin/perl ./sdriver.pl -t trace10.txt -s ./tsh -a -p
 4454 pts/135  S      0:00 ./tsh -p
 5205 pts/96   Ss+    0:00 /bin/bash --init-file /home/std/imhakyeum/.vscode-server/bin/1a5daa3a0231a0fbba4f14db7ec
cf99d7768e/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 5982 pts/125  Ss+    0:00 -bash
 6144 pts/104  Ss+    0:00 /bin/bash --init-file /home/std/kimyeonjin/.vscode-server/bin/af28b32d7e553898b2a91af498
b666fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 6707 pts/106  Ss+    0:00 -bash
 6715 pts/109  Ss+    0:00 -bash
 7525 pts/60   Ss+    0:00 /bin/bash --init-file /home/std/yen306/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 8590 pts/46   Ss+    0:00 -bash
 8817 pts/93   Ss+    0:00 -bash
```

rtest13의 결과와 동일한 결과를 출력했다.

```
bash-4.2$ make rtest13
./sdriver.pl -t trace13.txt -s ./tshref -a "-p"
#
# trace13.txt - Restart every stopped process in process group
#
tsh> ./mysplit 4
Job [1] (21825) stopped by signal 20
tsh> jobs
[1] (21825) Stopped ./mysplit 4
tsh> /bin/ps a
  PID TTY      STAT   TIME COMMAND
 1048 pts/167  Ss+    0:00 -bash
 1277 pts/27   Ss+    0:00 /bin/bash --init-file /home/std/leejiwon1125/.vscode-server/bin/ee2b180d582a7f601fa6ecfdad8
d9fd269ab1884/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 1567 tty1     Ss+    0:00 -bash
 1926 pts/142  Ss+    0:00 /bin/bash --init-file /home/std/kihyun/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666
fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 2184 pts/149  Ss+    0:00 -bash
 2291 pts/149  R      8:08 ./tsh -p
 3652 pts/66   Ss     0:00 /bin/bash --init-file /home/std/kkomy/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666f
debe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 3926 pts/116  S+     0:00 /tmp/.mount_nvim.atFiUct/usr/bin/nvim tsh.c
 4387 pts/150  S+     0:00 vim tsh.c
 4447 pts/135  T      0:00 make test10
 4451 pts/135  T      0:00 /usr/bin/perl ./sdriver.pl -t trace10.txt -s ./tsh -a -p
 4454 pts/135  S      0:00 ./tsh -p
 5205 pts/96   Ss+    0:00 /bin/bash --init-file /home/std/imhakyeum/.vscode-server/bin/1a5daa3a0231a0fbba4f14db7ec463
cf99d7768e/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 5982 pts/125  Ss+    0:00 -bash
 6144 pts/104  Ss+    0:00 /bin/bash --init-file /home/std/kimyeonjin/.vscode-server/bin/af28b32d7e553898b2a91af498b1f
b666fdebe0c/out/vs/workbench/contrib/terminal/browser/media/shellIntegration-bash.sh
 6707 pts/106  Ss+    0:00 -bash
 6715 pts/109  Ss     0:00 -bash
 7525 pts/60   Ss+    0:00 /bin/bash --init-file /home/std/yen306/.vscode-server/bin/af28b32d7e553898b2a91af498b1fb666
```

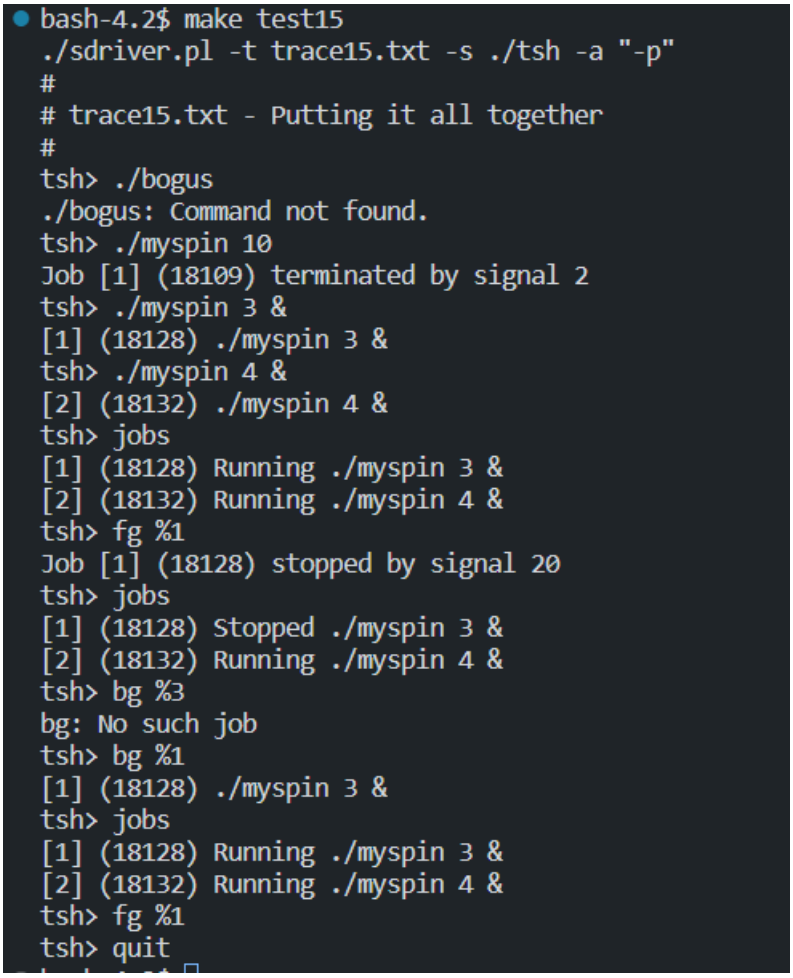
test14


bash-4.2$ make test14
./sdriver.pl -t trace14.txt -s ./tsh -a "-p"
#
trace14.txt - Simple error handling
#
tsh> ./bogus
./bogus: Command not found.
tsh> ./myspin 4 &
[1] (15992) ./myspin 4 &
tsh> fg
fg command requires PID or %jobid argument
tsh> bg
bg command requires PID or %jobid argument
tsh> fg a
fg: argument must be a PID or %jobid
tsh> bg a
bg: argument must be a PID or %jobid
tsh> fg 9999999
(9999999): No such process
tsh> bg 9999999
(9999999): No such process
tsh> fg %2
fg: No such job
tsh> fg %1
Job [1] (15992) stopped by signal 20
tsh> bg %2
bg: No such job
tsh> bg %1
[1] (15992) ./myspin 4 &
tsh> jobs
[1] (15992) Running ./myspin 4 &

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace14.txt -s ./tsh -a "-p"
#
# trace14.txt - Simple error handling
#
tsh> ./bogus
./bogus: Command not found
tsh> ./myspin 4 &
[1] (26326) ./myspin 4 &
tsh> fg
fg command requires PID or %jobid argument
tsh> bg
bg command requires PID or %jobid argument
tsh> fg a
fg: argument must be a PID or %jobid
tsh> bg a
bg: argument must be a PID or %jobid
tsh> fg 9999999
(9999999): No such process
tsh> bg 9999999
(9999999): No such process
tsh> fg %2
%2: No such job
tsh> fg %1
Job [1] (26326) stopped by signal 20
tsh> bg %2
%2: No such job
tsh> bg %1
[1] (26326) ./myspin 4 &
tsh> jobs
[1] (26326) Running ./myspin 4 &
```

test15


bash-4.2$ make test15
./sdriver.pl -t trace15.txt -s ./tsh -a "-p"
#
trace15.txt - Putting it all together
#
tsh> ./bogus
./bogus: Command not found.
tsh> ./myspin 10
Job [1] (18109) terminated by signal 2
tsh> ./myspin 3 &
[1] (18128) ./myspin 3 &
tsh> ./myspin 4 &
[2] (18132) ./myspin 4 &
tsh> jobs
[1] (18128) Running ./myspin 3 &
[2] (18132) Running ./myspin 4 &
tsh> fg %1
Job [1] (18128) stopped by signal 20
tsh> jobs
[1] (18128) Stopped ./myspin 3 &
[2] (18132) Running ./myspin 4 &
tsh> bg %3
bg: No such job
tsh> bg %1
[1] (18128) ./myspin 3 &
tsh> jobs
[1] (18128) Running ./myspin 3 &
[2] (18132) Running ./myspin 4 &
tsh> fg %1
tsh> quit

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace15.txt -s ./tsh -a "-p"
#
# trace15.txt - Putting it all together
#
tsh> ./bogus
./bogus: Command not found
tsh> ./myspin 10
Job [1] (26343) terminated by signal 2
tsh> ./myspin 3 &
[1] (26345) ./myspin 3 &
tsh> ./myspin 4 &
[2] (26347) ./myspin 4 &
tsh> jobs
[1] (26345) Running ./myspin 3 &
[2] (26347) Running ./myspin 4 &
tsh> fg %1
Job [1] (26345) stopped by signal 20
tsh> jobs
[1] (26345) Stopped ./myspin 3 &
[2] (26347) Running ./myspin 4 &
tsh> bg %3
%3: No such job
tsh> bg %1
[1] (26345) ./myspin 3 &
tsh> jobs
[1] (26345) Running ./myspin 3 &
[2] (26347) Running ./myspin 4 &
tsh> fg %1
tsh> quit
```

test16

```
bash-4.2$ make test16
./sdriver.pl -t trace16.txt -s ./tsh -a "-p"
#
# trace16.txt - Tests whether the shell can handle SIGTSTP and SIGINT
#     signals that come from other processes instead of the terminal.
#
tsh> ./mystop 2
Job [1] (18395) stopped by signal 20
tsh> jobs
[1] (18395) Stopped ./mystop 2
tsh> ./myint 2
Job [2] (18437) terminated by signal 2
```

treshref.out과 동일한 출력을 만들어냈다.

```
./sdriver.pl -t trace16.txt -s ./tsh -a "-p"
#
# trace16.txt - Tests whether the shell can handle SIGTSTP and SIGINT
#     signals that come from other processes instead of the terminal.
#
tsh> ./mystop 2
Job [1] (26359) stopped by signal 20
tsh> jobs
[1] (26359) Stopped ./mystop 2
tsh> ./myint 2
Job [2] (26362) terminated by signal 2
```